МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

КУРСОВАЯ РАБОТА

по дисциплине «Проектирование цифровых устройств»

Тема: Синтезатор

Студент гр. 1302 _____ Новиков Г.В.

Преподаватель _____ Каримов Т.И.

Санкт-Петербург

2024

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Новиков Г.В.

Группа 1302

Тема работы: Синтезатор

Исходные данные:
Должна быть использована Arduino, произведена разводка платы с помощью EasyEDA и корпус для нее.

Содержание пояснительной записки:
«Содержание», «Введение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 10.03.2024

Дата сдачи реферата: 04.07.2024

Дата защиты реферата: 04.07.2024

Студент _____ Новиков Г.В.

Преподаватель _____ Каримов Т.И.

АННОТАЦИЯ

В курсовой работе был спроектирован генератор звукового сигнала с возможностью управления высотой звука. Синтезатор имеет 4 режима работы и возможность использовать эффект вибрато. Была разведена плата в программе EasyEDA и создана трехмерная модель корпуса. Прототип имеет физическую реализацию на breadboard.

SUMMARY

In the course work, a sound signal generator with the ability to control the pitch of the sound was designed. The synthesizer has 4 operating modes and the ability to use the vibrato effect. The board was laid out in the EasyEDA program and a three-dimensional model of the case was created. The prototype has a physical implementation on the breadboard.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы состоит в проектировании и физической реализации синтезатора, разводке платы и создании трехмерной модели корпуса. Для реализации проекта была использована Arduino Pro Micro.

ФОРМУЛИРОВКА ЗАДАНИЯ

Реализовать синтезатор на платформе Arduino с возможностью управления высотой звукового сигнала, несколькими режимами работы, эффектом вибрато с возможностью его настройки. Создать модель разведенной платы в программе EasyEDA и трехмерную модель корпуса.

СХЕМА ПРОЕКТА

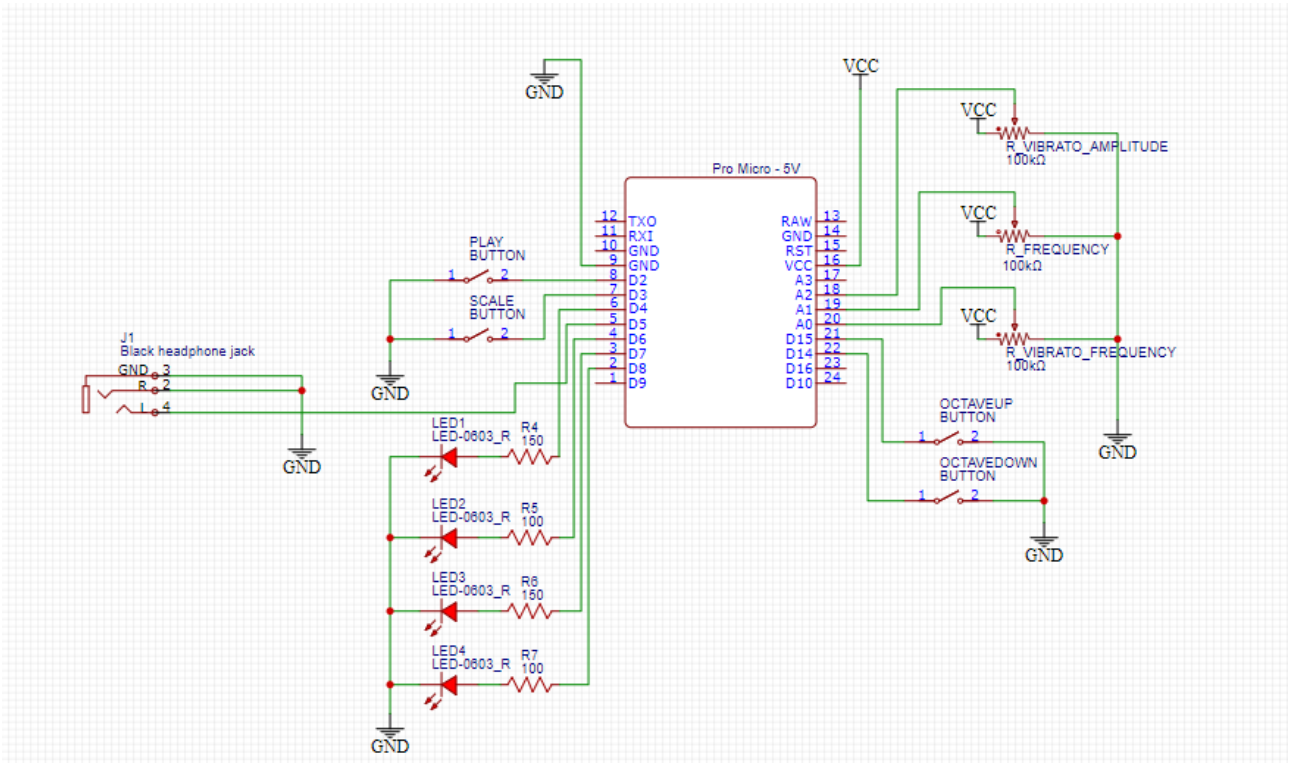

Рис. 1. Схема, реализованная в EasyEDA

ОПИСАНИЕ СОСТАВНЫХ ЧАСТЕЙ ПРОЕКТА И ВЫПОЛНЯЕМЫХ ИМИ ФУНКЦИЙ

Используемые в проекте детали: Arduino Pro Micro, 3 потенциометра, 4 кнопки, разъем jack 3.5, 4 светодиода и 4 резистора к ним, breadboard и провода.

Arduino Pro Micro отвечает за взаимодействие всех компонентов. Она реализует считывание данных с потенциометров, отслеживает нажатие кнопок, генерирует звуковой сигнал с помощью функции tone и выводит его. Для контроля дребезга кнопок используется библиотека “InputDebounce”. Функция tone генерирует прямоугольную "волну", заданной частоты с 50% рабочим циклом. Частота задается с помощью потенциометра, диапазон при управлении только потенциометром – от ноты С до С следующей октавы. Для изменения октавы используются 2 кнопки. Также Arduino отвечает за смену режимов работы по кнопке и включение соответствующего светодиода. Реализовано 4 режима работы, режим определяет распределение частот в зависимости от положения потенциометра, контролирующего высоту звука. Режим 1 – нет привязки к нотам, потенциометр задает частоту напрямую. Режим 2 – используются 12 нот (хроматическая гамма) и нота С следующей октавы. Режим 3 – мажорный лад и нота С следующей октавы. Режим 4 – минорный лад и нота С следующей октавы. Для работы проекта достаточно 5В питания, подключаемого к Arduino.

В проекте используются 4 кнопки:

1. Кнопка 1: если нажата звук выводится, если не нажата – не выводится
2. Кнопка 2 и 3: увеличение и уменьшение текущей октавы
3. Кнопка 4: смена режима работы

В проекте используются 3 потенциометра:

1. Потенциометр 1: отвечает за высоту звука
2. Потенциометр 2: громкость эффекта вибрато
3. Потенциометр 3: частота колебаний вибрато

Эффект вибрато реализован следующим образом: отдельно от основного сигнала генерируется синусоидальная волна с заданными потенциометрами

параметрами, она умножается на высоту звука и прибавляется к основной частоте.

Светодиоды используются для обозначения текущего режима.

Разъем jack 3.5 нужен для вывода звука.

РАЗВЕДЕНИЕ ПЛАТЫ И КОРПУС

Разведенная плата:

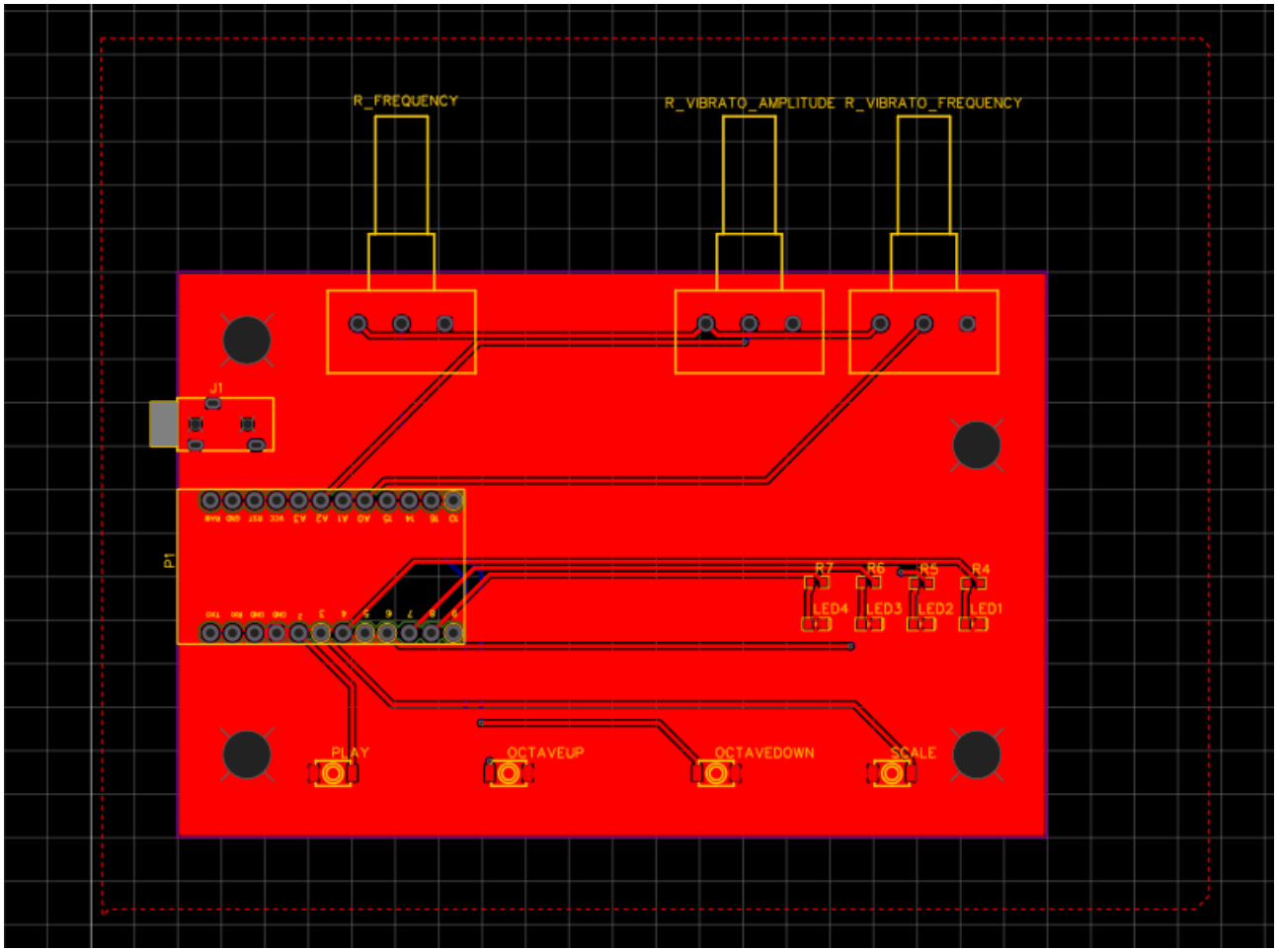

Рис. 2. Верхний слой

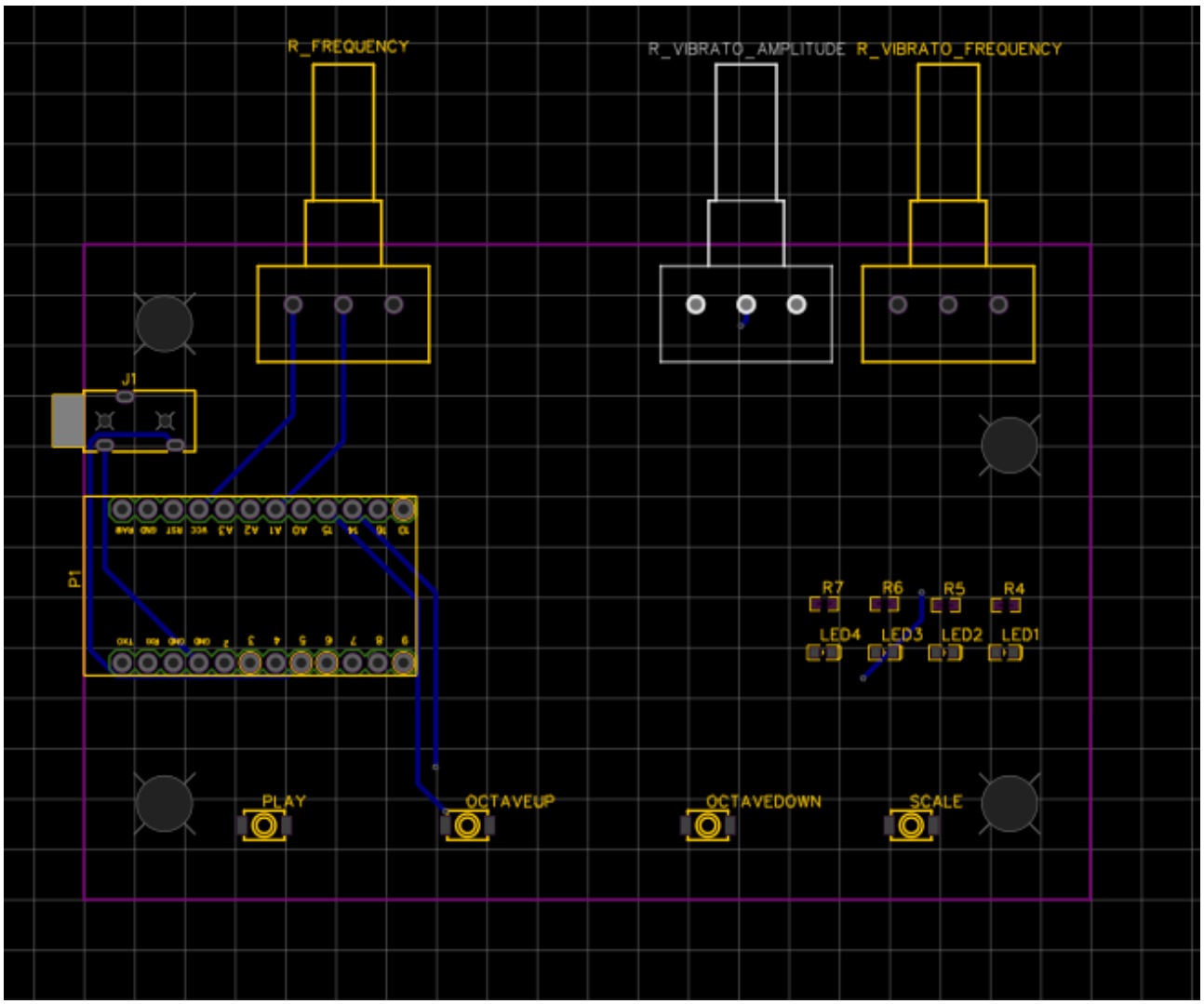

Рис. 3. Без верхнего слоя

Размеры 65x165 мм.

Отверстия сделаны для того, чтобы можно было соединить части корпуса.

3D-модель платы:

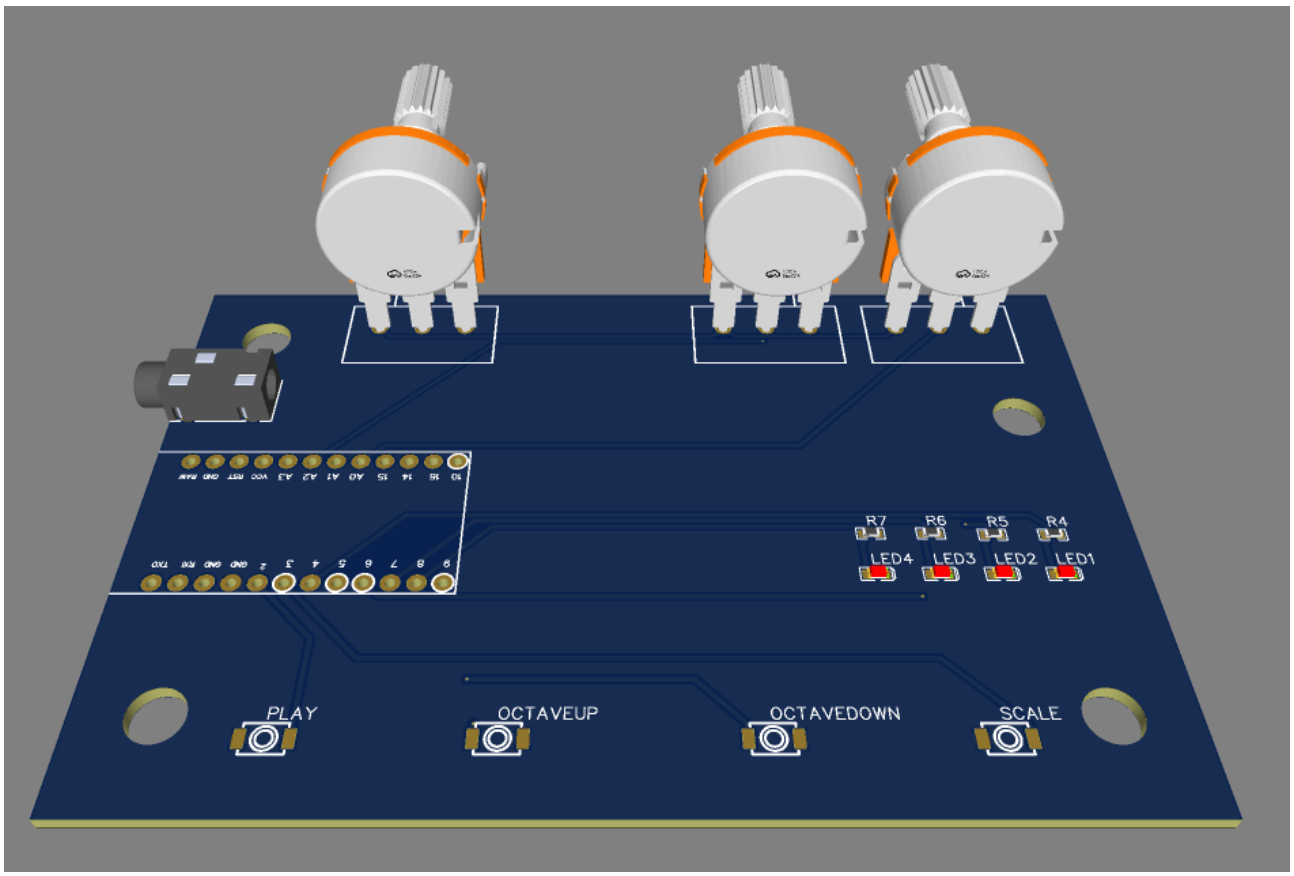

Рис. 4. 3D

Под плату был спроектирован корпус в программе КОМПАС-3D. Состоит он из 2 деталей, которые крепятся между собой с помощью винтов М4. Разведенная плата крепится на винты М2 длиной 3мм.

Основной корпус (сборка):

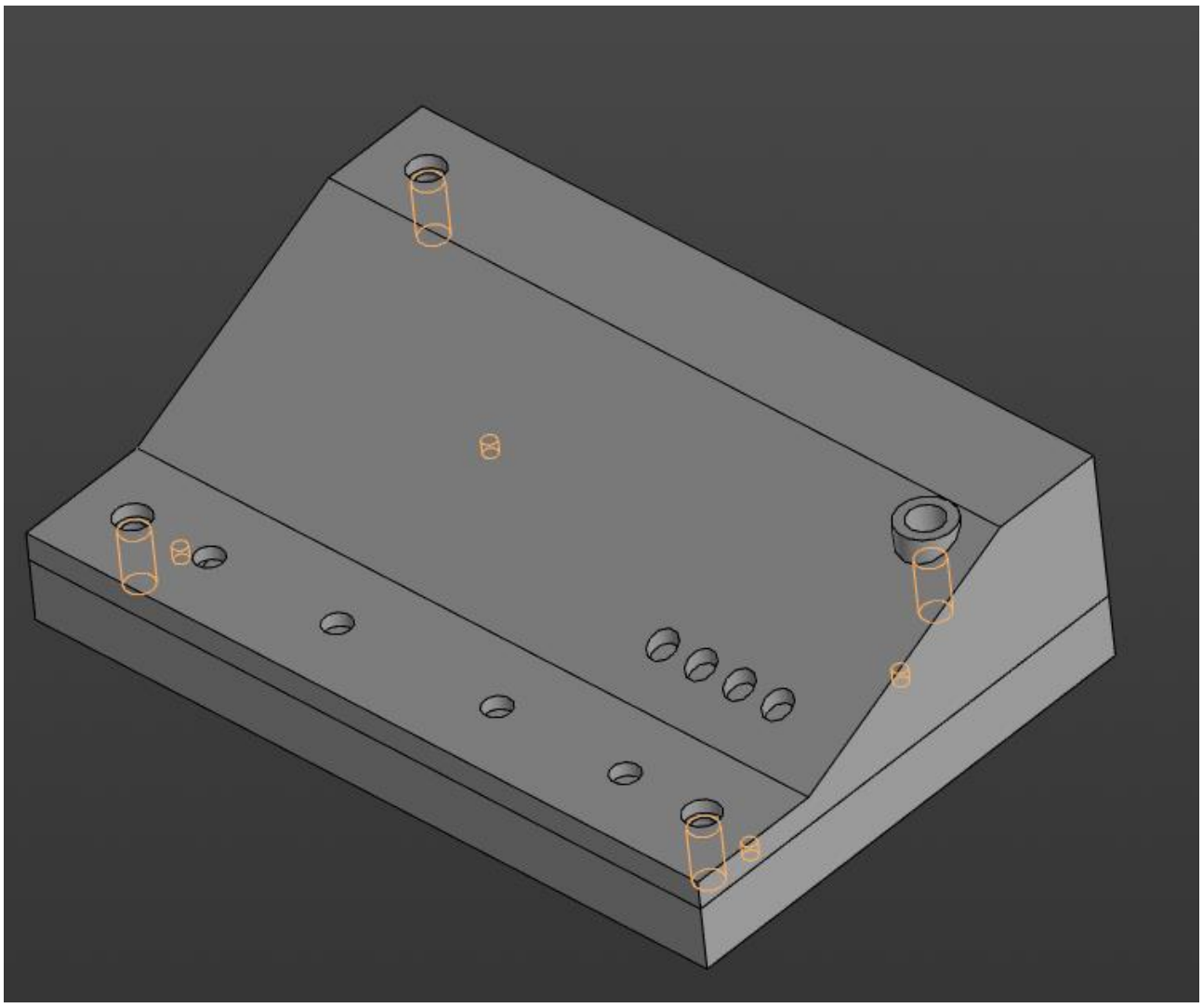

Рис. 5. Основной корпус

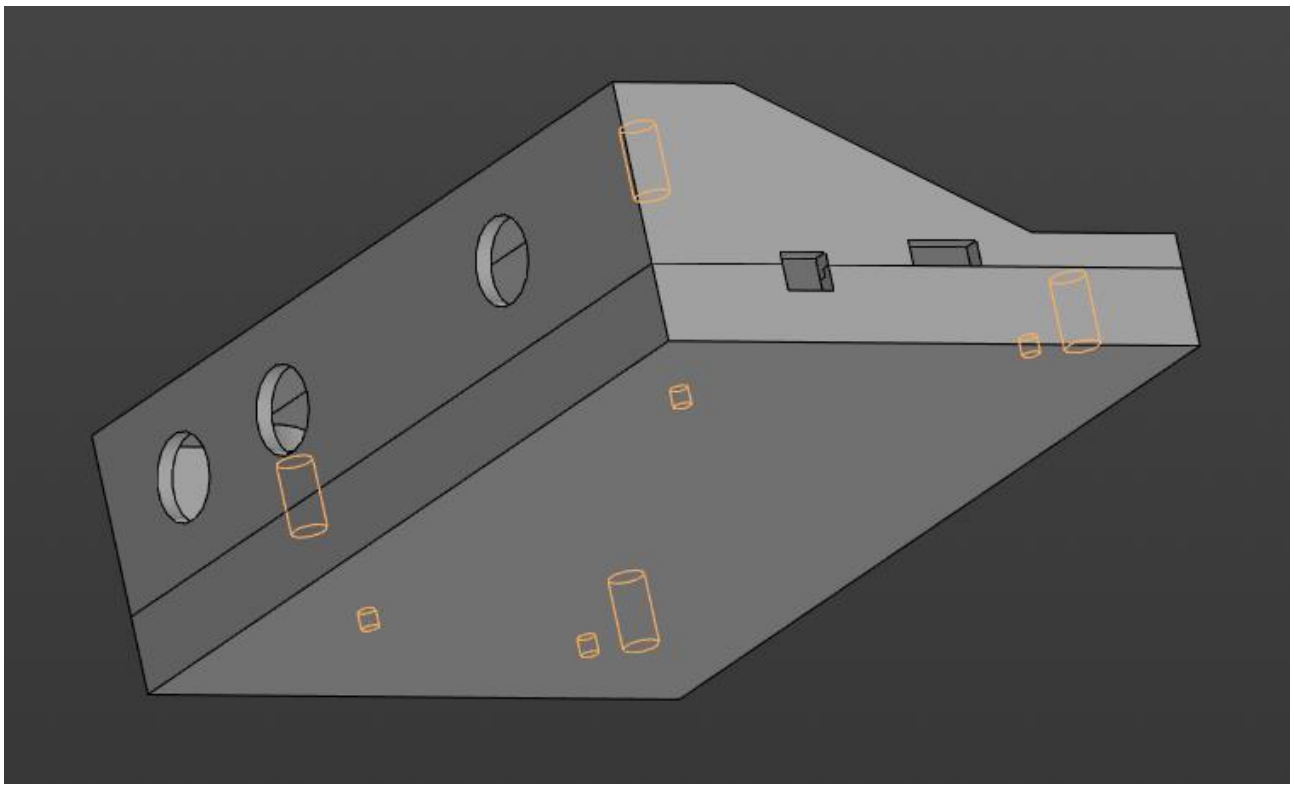

Рис. 6. Основной корпус

В сборку входят 2 детали – нижняя часть и верхняя. К нижней крепится разведенная плата, затем вставляются потенциометры в отверстия, после чего части можно соединить.

Части корпуса:

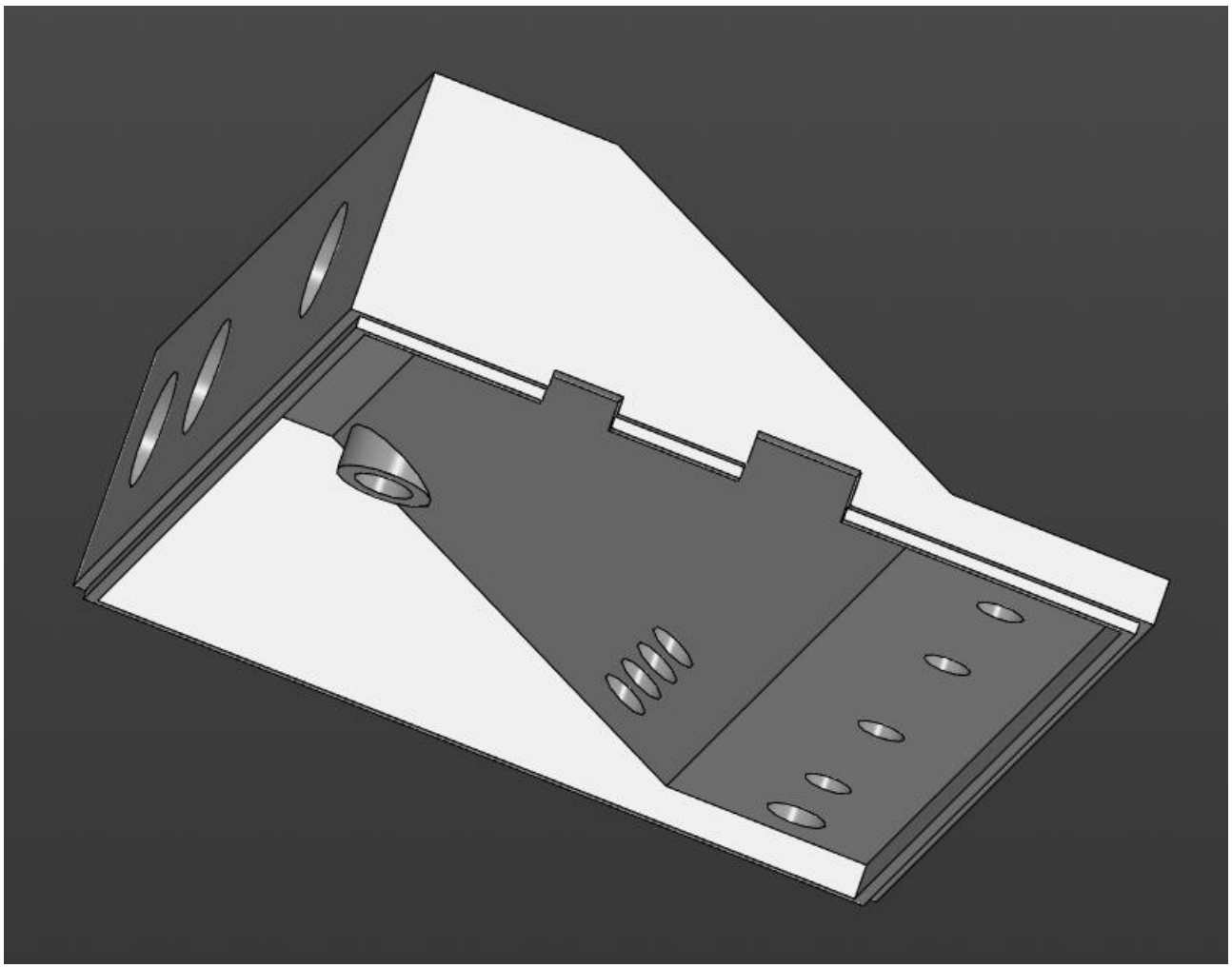

Рис. 7. Верхняя часть

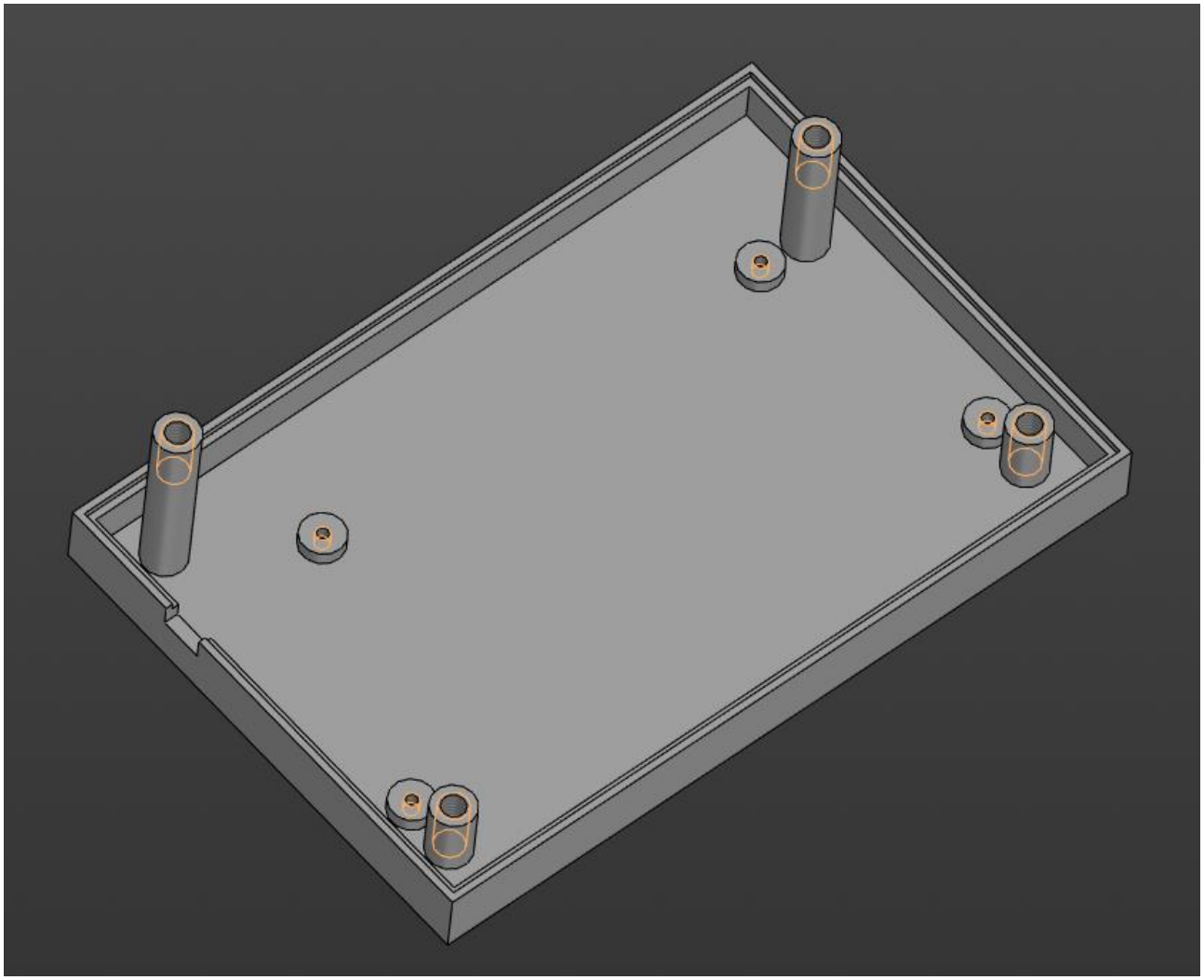

Рис. 8. Нижняя часть

ФОТОГРАФИИ ПРОЕКТА

Физический макет на breadboard:

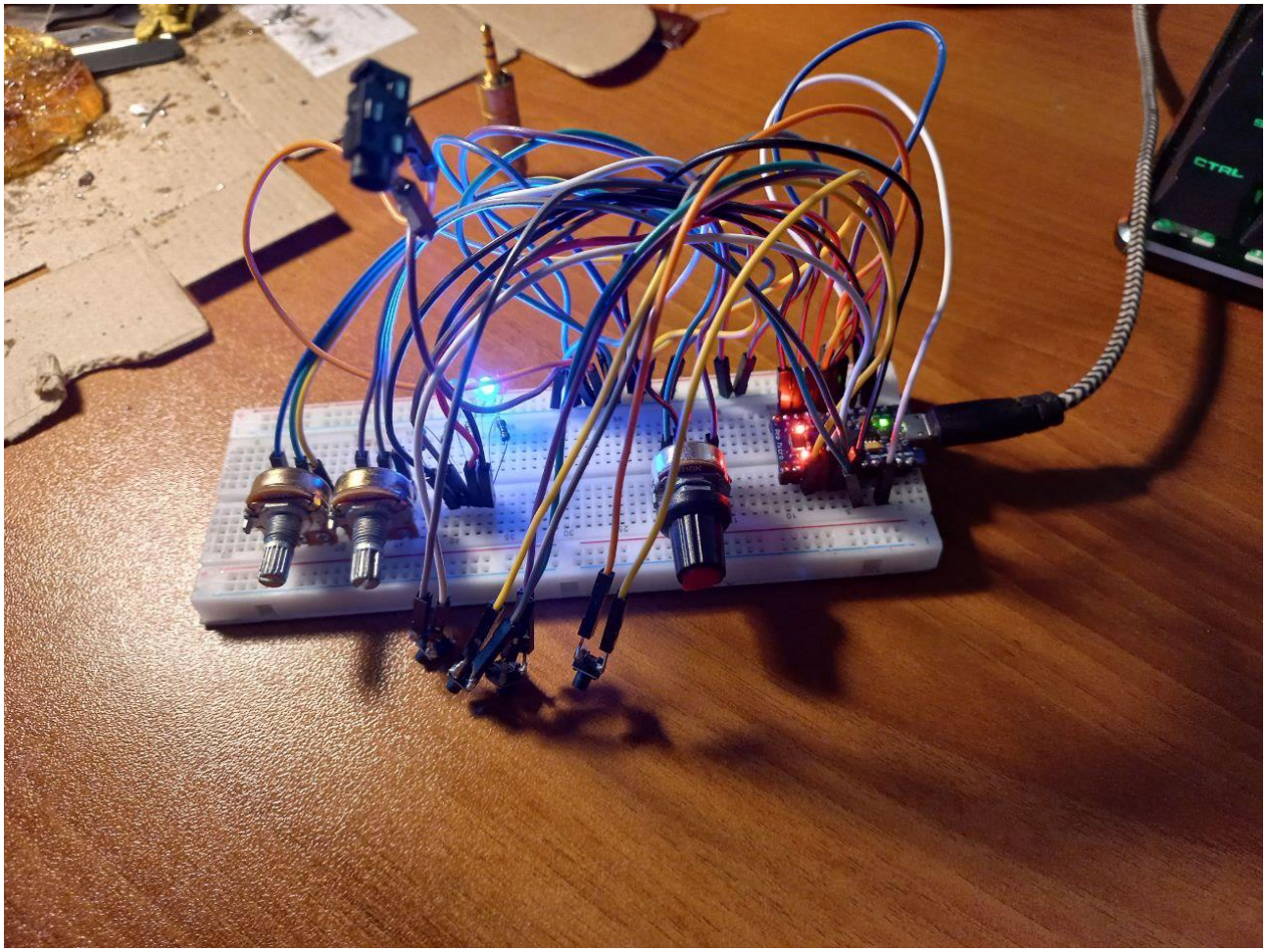

Рис. 9. Прототип

ЛИСТИНГ ПРОШИВКИ КОНТРОЛЛЕРА

```
#include "InputDebounce.h"


#define BUTTON_DEBOUNCE_DELAY 20    // [ms]


const int out_pin = 5;

const int pot1 = A1;

const int pot2 = A0;

const int pot3 = A2;

const int pin_on_off = 2;

const int pin_plus_octave = 15;

const int pin_minus_octave = 14;

const int pin_change_scale = 3;

const int led_pins[] = { 4, 6, 7, 8 };


InputDebounce button_on_off;

InputDebounce button_plus_octave;

InputDebounce button_minus_octave;

InputDebounce button_change_scale;


const int MIN_FREQUENCY = 130;

const int MAX_FREQUENCY = 4187;

const int A_NOTE = 440;

int frequency;


const int MIN_OCTAVE = -2;

const int MAX_OCTAVE = 2;

int octave = -1;


enum Scale { None, Chromatic, Major, Minor };

Scale scale;
```

```
int major_note_numbers[] = { 0, 2, 4, 5, 7, 9, 11, 12 };

int minor_note_numbers[] = { 0, 2, 3, 5, 7, 8, 10, 12 };


float vibrato_amplitude = 0.01;

float vibrato_frequency = 2;

float vibrato_x = 0;

int vibrato_time = 0;


float vibrato_wave()

{

   int new_vibrato_time = millis();

   vibrato_x += (new_vibrato_time - vibrato_time) * 6.28 * vibrato_frequency / 1000.0;

   vibrato_time = new_vibrato_time;


   return random(700, 1300) / 1000.0 * vibrato_amplitude * sin(vibrato_x);


}


void butt_change_scale_callback(uint8_t pinIn)

{

   digitalWrite(led_pins[scale], LOW);

   scale = (Scale)((scale + 1) % 4);

   digitalWrite(led_pins[scale], HIGH);

}


void butt_minus_octave_pressed_callback(uint8_t pinIn)

{

   if (octave > MIN_OCTAVE) octave--;

}


void butt_plus_octave_pressed_callback(uint8_t pinIn)

{
```

```
    if (octave < MAX_OCTAVE) octave++;
}

void butt_on_off_pressed_callback(uint8_t pinIn)
{
    tone(out_pin, frequency);
}

void butt_on_off_released_callback(uint8_t pinIn)
{
    noTone(out_pin);
}

int get_note_pitch(int pot_value)
{
    float h = 12 * octave + 3.0;  // C note of selected octave
    switch (scale)
    {
        case None:
            h += pot_value / 1024.0 * 12.0;
            break;
        case Chromatic:
            h += constrain(map(pot_value, 0, 1023, 0, 13), 0, 12);
            break;
        case Major:
            h += major_note_numbers[constrain(map(pot_value, 0, 1023, 0, 8), 0, 7)];
            break;
        case Minor:
            h += minor_note_numbers[constrain(map(pot_value, 0, 1023, 0, 8), 0, 7)];
            break;
        default:
            break;
```

```
    }

    float pitch = pow(2, h / 12.0) * A_NOTE;
    pitch += vibrato_wave() * (float)pitch;
    return pitch;
}

float get_vibrato_frequency(int pot_value)
{
    return map(pot_value, 0, 1023, 10, 100) / 10.0;
}

float get_vibrato_amplitude(int pot_value)
{
    return pot_value / 1023.0 * 0.02;
}

void setup() {
    // put your setup code here, to run once:
    pinMode(out_pin, OUTPUT);
    for (int i = 0; i < 4; i++)
    {
        pinMode(led_pins[i], OUTPUT);
    }
    scale = None;
    digitalWrite(led_pins[scale], HIGH);

    button_on_off.registerCallbacks(butt_on_off_pressed_callback, butt_on_off_released_callback, butt_on_off_pressed_callback, NULL);
    button_plus_octave.registerCallbacks(butt_plus_octave_pressed_callback, NULL, NULL, NULL);
    button_minus_octave.registerCallbacks(butt_minus_octave_pressed_callback, NULL, NULL, NULL);
    button_change_scale.registerCallbacks(butt_change_scale_callback, NULL, NULL, NULL);
```

```
    button_on_off.setup(pin_on_off, BUTTON_DEBOUNCE_DELAY, InputDebounce::PIM_INT_PULL_UP_RES);
    button_plus_octave.setup(pin_plus_octave, BUTTON_DEBOUNCE_DELAY, InputDebounce::PIM_INT_PULL_UP_RES);
    button_minus_octave.setup(pin_minus_octave, BUTTON_DEBOUNCE_DELAY, InputDebounce::PIM_INT_PULL_UP_RES);
    button_change_scale.setup(pin_change_scale, BUTTON_DEBOUNCE_DELAY, InputDebounce::PIM_INT_PULL_UP_RES);

    pinMode(pot1, INPUT);
    pinMode(pot2, INPUT);
    pinMode(pot3, INPUT);

    Serial.begin(200000);
}

void loop() {
    // put your main code here, to run repeatedly:
    unsigned long now = millis();

    button_on_off.process(now);
    button_plus_octave.process(now);
    button_minus_octave.process(now);
    button_change_scale.process(now);

    frequency = get_note_pitch(analogRead(pot1));
    vibrato_frequency = get_vibrato_frequency(analogRead(pot2));
    vibrato_amplitude = get_vibrato_amplitude(analogRead(pot3));
    Serial.print(octave);
    Serial.print(" ");
    Serial.print(scale);
    Serial.print(" ");
    Serial.print(vibrato_frequency);
    Serial.print(" ");
    Serial.println(frequency);
}
```

ЗАКЛЮЧЕНИЕ

После выполнения курсовой работы у нас есть рабочий физический прототип синтезатора, разведенная плата и смоделированный трехмерный корпус. Поставленные в начале работы цели были реализованы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Arduino URL: https://docs.arduino.cc/ (дата обращения 19.05.24)
2. Документация библиотеки InputDebounce URL: https://github.com/Mokolea/InputDebounce (дата обращения 25.05.24)